Shallow Semantics for Relation Extraction

Sanda Harabagiu, Cosmin Adrian Bejan and **Paul Morărescu**
Human Language Technology Research Institute
University of Texas at Dallas
Richardson, TX 75083-0688, USA
sanda,ady,paul@hlt.utdallas.edu

Abstract

This paper presents a new method for extracting meaningful relations from unstructured natural language sources. The method is based on information made available by shallow semantic parsers. Semantic information was used (1) to enhance a dependency tree kernel; and (2) to build semantic dependency structures used for enhanced relation extraction for several semantic classifiers. In our experiments the quality of the extracted relations surpassed the results of kernel-based models employing only semantic class information.

1 Introduction

With the advent of the Internet, more and more information is available electronically. Most of the time, information on the Internet is unstructured, generated in textual form. One way of automatically identifying information of interest from the vast Internet resources is by recognizing relevant entities and meaningful relations they share. Examples of entities are PERSONS, ORGANIZATIONS and LOCATIONS. Examples of relations are ROLE, NEAR and SOCIAL.

In the 90s, the Message Understanding Conferences (MUCs) and the TIPSTER programs gave great impetus to research in information extraction (IE). The systems that participated in the MUCs have been quite successful at recognizing relevant entities, reaching near-human precision with over 90% accuracy. More recently, the Automatic Content Extraction (ACE) program focused on identifying not only relevant entities, but also meaningful relations between them. If success in recognizing entities with high precision was attributed to the usage of finite-state transducers, in the last three years the dominant successful technique for extracting relations was based on kernel methods.

Kernel methods are non-parametric density estimation techniques that compute a *kernel function* to measure the similarity between data instances. [8] introduced the formalization of relation extraction in terms of tree kernels, which are kernels that take advantage of syntactic trees. [1] extended the work by using dependency trees that describe the grammatical relations between the words of a sentence. Furthermore, each word of a dependency tree is augmented with lexico-semantic information, including semantic information available from the WordNet database [2]. The average precision of relation extraction using dependency trees was reported in [1] to be 67.5%. Since meaningful relations between relevant entities are of semantic nature, we argue that additional semantic resources should be used for extracting relations from texts. In this work, we were interested in investigating the contribution of two shallow semantic parsing techniques to the quality of relation extraction.

We explored two main resources: *PropBank* and *FrameNet*. Proposition Bank or PropBank is a one million word corpus annotated with predicate-argument structures. The corpus consists of the Penn Treebank 2 Wall Street Journal texts (*www.cis.upenn.edu/∼treebank*). The PropBank annotations were performed at University of Pennsylvania (*www.cis.upenn.edu/∼ace*). To date PropBank has addressed only predicates lexicalized by verbs, proceeding from the most to the least common verbs while annotating verb predicates in the corpus. The FrameNet project (*www.icsi.berkeley.edu/∼framenet*) produced a lexico-semantic resource encoding a set of *frames*, which represent schematic representations of situations characterized by a set of *target words*, or lexicalized predicates, which can be verbs, nouns or adjectives. In each frame, various participants and conceptual roles are related by case-roles or theta-roles which are called *frame elements* or FEs. FEs are local to each frame, some are quite general while others are specific to a small family of lexical items. FrameNet annotations were performed on a corpus of over three million words. Recently, semantic parsers using PropBank and FrameNet have started to become available. In each sentence, verbal or nominal predicates are discovered in relation to their arguments or FEs. Our investigation shows that predicate-arguments structures and semantic frames discovered by shallow semantic parsers play an important role in discovering extraction relations. This is due to the fact that arguments of extracted relations belong to arguments of predicates or to FEs.

To investigate the role of semantic information for relation extraction we have used two shallow semantic parsers, one trained on PropBank and one on FrameNet. We used the semantic information identified by the parsers in two ways. First it was used to enhance the features of dependency kernels. Second, it was used to generate a new representation, called semantic dependency structure. The results of the ex-

periments indicate that not only the precision of kernel methods is improved, but also relation extraction based on shallow semantics outperforms kernel-based methods.

The remainder of this paper is organized as follows. Section 2 describes the semantic parsers. Section 3 details the kernel methods and their enhancement with semantic information. In Section 4 we show how relation extraction based on semantic information is produced. Section 5 details the experimental results and Section 6 summarizes our conclusions.

2 Shallow Semantic Parsing

Shallow semantic information represented by predicates and their arguments, or frames and their FEs, can be identified in text sentences by semantic parsers. The idea of automatically identifying and labeling shallow semantic information was pioneered by [3]. Semantic parsers operate on the output of a syntactic parser. When using the PropBank information, the semantic parser (1) identifies each verbal predicate and (2) labels its arguments. The expected arguments of a predicate are numbered sequentially from Arg0 to Arg5. Additionally, the arguments may include functional tags from Treebank, e.g. ArgM-DIR indicates a directional, ArgM-LOC indicates a locative and ArgM-TMP stands for a temporal. When using the FrameNet information, the semantic parser (1) identifies the target words; (2) disambiguates the semantic frame for each target word; and (3) labels the FEs that relate to the target word based on the frame definition. For example, in FrameNet the EMPLOYMENT frame has the FEs: Employee, Employer, Purpose, Compensation, Manner, Place, Position, Time, Field, Duration and Task.

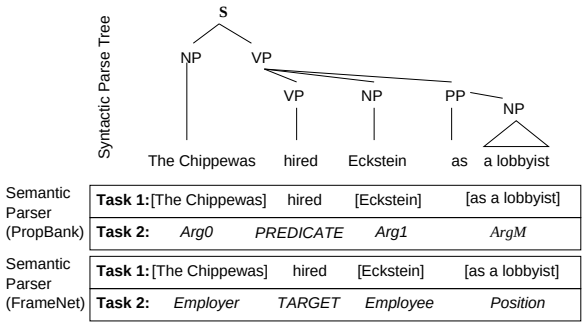

Figure 1: Sentence with labeled semantic roles.

Figure 1 illustrates the output of both semantic parsers when processing a sentence. The parsing consists of two tasks: (1) identifying the parse tree constituents corresponding to arguments of each predicate or FEs of each frame; and (2) recognizing the role corresponding to each argument or FE. Each task is cast as a separate classifier. For example, task 1 identifies the two NPs and the PP as arguments and FEs respectively. The second classifier assigns the specific roles: Arg0, Arg1 and ArgM given the predicate "hired" and FEs *Employer*, *Employee* and *Position* given the target word "hired". In the case of semantic parsing parsed on FrameNet, a third task of finding the frame corresponding to the sentence is cast as a third classification problem.

− PHRASE TYPE (pt): This feature indicates the syntactic type of the phrase labeled as a frame element, e.g. NP for Employer in Figure 1.
− PARSE TREE PATH (path): This feature contains the path in the parse tree between the predicate phrase and the target word, expressed as a sequence of nonterminal labels linked by direction symbols (up or down), e.g. NP ↑S ↓VP ↓VP for Employer in Figure 1.
− POSITION (pos) − Indicates if the constituent appears before or after the predicate in the sentence.
− VOICE (voice) − This feature distinguishes between active or passive voice for the predicate phrase.
− HEAD WORD (hw) − This feature contains the head word of the evaluated phrase. Case and morphological information are preserved.
− GOVERNING CATEGORY (gov) − This feature applies to noun phrases only, and it indicates if the NP is dominated by a sentence phrase (typical for subject arguments with active−voice predicates), or by a verb phrase (typical for object arguments).
− TARGET WORD − In our implementation this feature consists of two components: (1) WORD: the word itself with the case and morphological information preserved; and (2) LEMMA which represents the target normalized to lower case and infinitive form for the verbs or singular for nouns.

Figure 2: Feature Set 1 (FS1).

− CONTENT WORD (cw) − Lexicalized feature that selects an informative word from the constituent, different from the head word.
PART OF SPEECH OF HEAD WORD (hPos) − The part of speech tag of the head word.
PART OF SPEECH OF CONTENT WORD (cPos) −The part of speech tag of the content word.
NAMED ENTITY CLASS OF CONTENT WORD (cNE) − The class of the named entity that includes the content word
BOOLEAN NAMED ENTITY FLAGS − A feature set comprising: − neOrganization: set to 1 if an organization is recognized in the phrase − neLocation: set to 1 a location is recognized in the phrase − nePerson: set to 1 if a person name is recognized in the phrase − neMoney: set to 1 if a currency expression is recognized in the phrase − nePercent: set to 1 if a percentage expression is recognized in the phrase − neTime: set to 1 if a time of day expression is recognized in the phrase − neDate: set to 1 if a date temporal expression is recognized in the phrase

Figure 3: Feature Set 2 (FS2).

Several classifiers were previously tried for this problem: decision trees [7], and Support Vector Machines (SVMs) [6]. Based on the results of the CoNLL and Senseval-3 evaluations[1], we selected an implementation based on SVMs, using the SVMlight package [2]. For the semantic parser based on PropBank, we combined feature sets: FS1 (illustrated in Figure 2) introduced in [3], FS2 (illustrated in Figure 3) introduced in [7], FS3 (illustrated in Figure 4) introduced in [6]. For the semantic parser based on FrameNet we have also added the feature set FS4 (illustrated in Figure 5). The se-

[1]The Conference on Natural Language Learning (CoNLL) (http://cnts.uia.ac.be/signll/shared.html) has focused in 2004 on the problem of semantic role labeling, using PropBank data. Senseval-3 (www.senseval.org/senseval3) had a semantic role labeling task, using FrameNet data.

[2]http://svmlight.joachims.org/

mantic parser using FrameNet also required the disambiguation of the frame. To disambiguate the frame, we used the BayesNet algorithm implemented in the Weka learning package [3]. The features that were used are (1) the target word; (2) the part-of-speech of the target word; (3) the phrase type of all the FEs (e.g. NP, VP, PP); and (4) the *grammatical function*. The grammatical function is defined in Figure 4. To learn the grammatical function, we again used the SVM with the features FS1, FS2, the features `Human` and `Target-Type` from FS3 and only the feature `PP-PREP` from FS4.

HUMAN: This feature indicates whether the syntactic phrase is either (1) a personal pronoun or (2) a hyponym of sense 1 of PERSON in WordNet
SUPPORT_VERBS that are recognized for adjective or noun target words have the role of predicate for the FEs. For example, if the target = clever, in the sentence "Smith is very clever, but he's no Einstein", the FE = Smith is an argument of the support verb 'is' rather than of the target word. The values of this feature are either (1) The POS of the head of the VP containing the target word or (2) NULL if the target word does not belong to a VP
TARGET−TYPE: the lexical class of the target word, e.g. VERB, NOUN or ADJECTIVE
LIST_CONSTITUENT (FEs): This feature represents a list of the syntactic consituents covering each FE of the frame recognized in a sentence. For the example illustrated in Figure 1, the list is: [NP, NP, PP]
Grammatical Function: This feature indicates whether the FE is: − an External Argument (Ext) − an Object (Obj) − a Complement (Comp) − a Modifier (Mod) − Head noun modified by attributive adjective (Head) − Genitive determiner (Gen) − Appositive (Appos)
LIST_Grammatical_Function: This feature represents a list of the grammatical functions of the FEs recognized in the sentence.
NUMBER_FEs: This feature indicates how many FEs were recognized in each sentence.
FRAME_NAME: This feature indicates the name of the semantic frame for which FEs are labeled
COVERAGE: This feature indicates whether there is a syntactic structure in the parse tree that perfectly covers the FE
CORE: This feature indicates whether the FE is one that instantiates a conceptually necessary participant of a frame. For example, in the REVENGE frame, Punishment is a core element. The values of this feature are: (1) core; (2) peripheral and (3) extrathemathic. FEs that mark notions such as Time, Place, Manner and Degree are peripheral. Extratematic FEs situate an event against a backdrop of another event, by evoking a larger frame for which the target event fills a role.
SUB_CORPUS: In FrameNet, sentences are annotated with the name of the subcorpus they belong to. For example, for a verb target word, V−swh represents a subcorpus in which the trget word is a predicate to a FE included in a relative clause headed by a wh−word.

Figure 4: Feature Set 3 (FS3).

3 Dependency Tree Kernels

In [1] the relation extraction problem was cast as a classification problem based on kernels that operate on dependency trees. Kernels measure the similarity between two instances of a relation. If $\mathbf{X}$ is the instance space, a kernel function is a mapping K:$\mathbf{X}\times\mathbf{X}\rightarrow[0,\infty)$ such that given two instances

[3] *http://www.cs.waikato.ac.nz/∼ml*

PARSE TREE PATH WITH UNIQUE DELIMITER − This feature removes the direction in the path, e.g. VBN−VP−ADVP
PARTIAL PATH − This feature uses only the path from the constituent to the lowest common ancestor of the predicate and the constituent
FIRST WORD − First word covered by constituent
FIRST POS − POS of first word covered by constituent
LAST WORD − Last word covered by the constituent
LAST POS − POS of last word covered by the constituent
LEFT CONSTITUENT − left sibling constituent label
LEFT HEAD − Left sibling head word
LEFT POS HEAD − Left sibling POS of head word
RIGHT CONSTITUENT − right sibling constituent label
RIGHT HEAD − Right sibling head word
RIGHT POS HEAD − Right sibling POS of head word
PP PREP − If constituent is labeled PP get first word in PP
DISTANCE − Distance in the tree from constituent to the target word

Figure 5: Feature Set 4 (FS4).

x and y, $K(x,y) = \sum_i \phi_i(x)\phi_i(y) = \phi(x)\cdot\phi(y)$, where $\phi_i(x)$ is some feature function over the instance x. The instances can be represented in several ways. First, each sentence where a relation of interest occurs can be viewed as a list of words. Thus, the similarity between two instances represented in this way is computed as the number of common words between the two instance sentences. All words from instances x and y are indexed and $\phi_i(x)$ is the number of times instance x contains the word referenced by i. Such a kernel is known as *bag-of-words kernel*. When sentences are represented as strings of words, *string kernels*, count the number of common subsequences in the two strings and weight their matches by their length. Thus $\phi_i(x)$ is the number of times string x contains the subsequence referenced by i. If the instances are represented by syntactic trees, more complex kernels are needed. A class of kernels, called *convolution kernels*, was proposed to handle such instance representations. Convolution kernels measure the similarity between two structured instances by summing the similarity of their substructures. Thus, given all possible substructures in instances x and y, $\phi_i(x)$ counts not only the number of times the substructure referenced by i is matched into x, but also how many times it is matched into any of its substructures.

Given a training set T=$\{x_1,\ldots x_N\}$, kernel methods compute the *Gram* matrix G such that G_{ij}=$K(x_i,x_j)$. G enables a classifier to find a hyperplane which separates instances of different classes. G enables classifiers to find a separating hyperplane that separates positive and negative examples. When a new instance y needs to be classified, y is projected into the feature space defined by the kernel function. Classification consists of determining on which side of the separating hyperplane y lies. *Support Vector Machines*(SVMs) formulate the task of finding the separating hyperplane as a solution to a quadratic programming problem.Therefore, following the solution proposed by [1], they are used for classifying relation instances in texts.

To measure the similarity between two instances of the

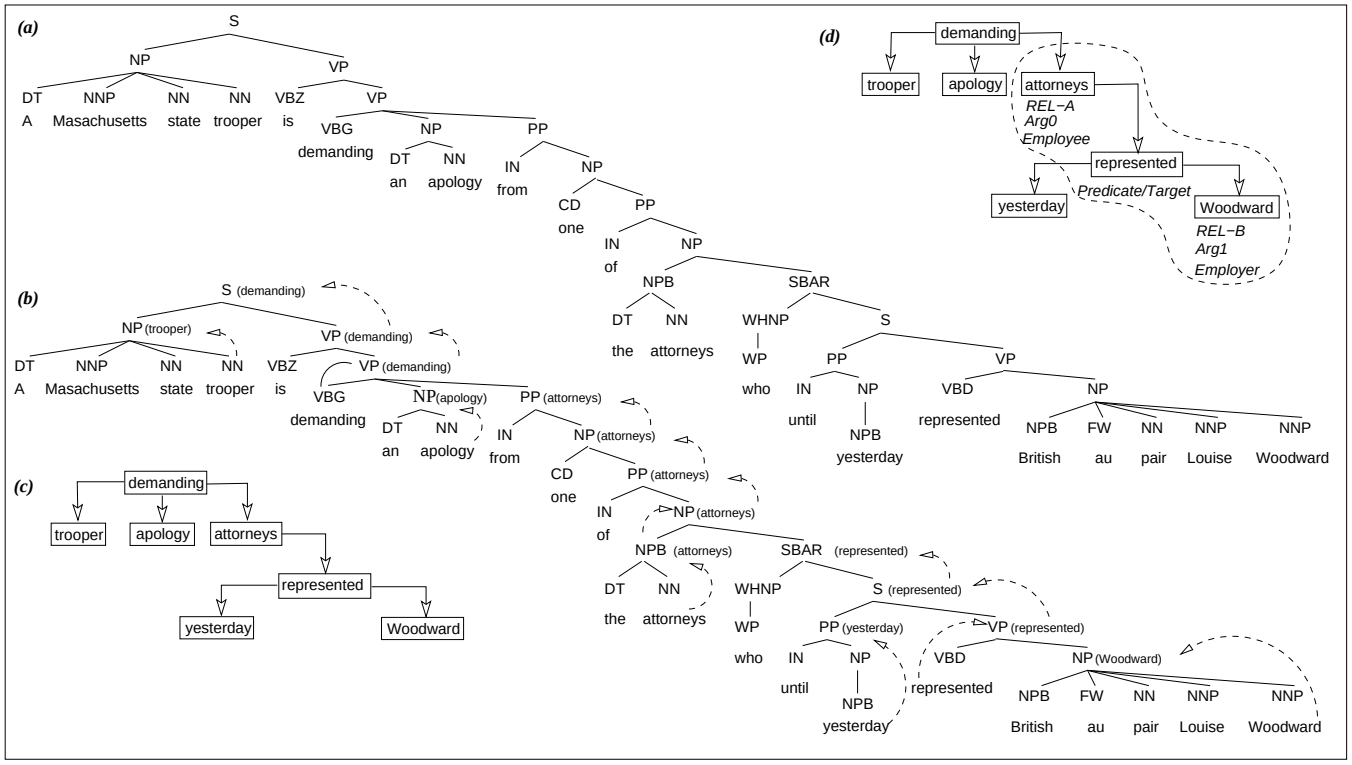

Figure 6: (a) Syntactic parse; (b) Head word propagation; (c) Dependency tree; (d) Semantic dependency tree.

same extraction relation both [1] and [8] relied on kernels that operate on trees expressing syntactic information. To build such a tree we have used the Collins syntactic parser. When using this parser, for each constituent in the parse tree, we also have access to a *dependency* model that enables the mapping of parse trees into sets of binary relations between the head-word of each component and its sibling words. For example, Figure 6(a) describes the parse tree of a sentence. For each possible constituent in the parse tree, rules first described in [4] identify the head-child and propagate the head word to the parent. Figure 6(b) illustrates the propagation for the parse tree illustrated in Figure 6(a). When the propagation is over, head-modifier relations are extracted, generating a dependency structure. Figure 6(c) illustrates the dependency structure of the sentence that was analyzed syntactically in Figure 6(a). The nodes of this dependency structure were augmented with features, to enable the calculation of the kernel. Figure 7 lists the features that were assigned to each node in the dependency tree. Two sets of features were used: F1, the features proposed in [1] and F2, a new set of features that we have added. Feature set F1 contains: (1) the word; (2) the part-of-speech (POS) (24 values); (3) a generalized POS (5 values); (4) the tag of the nonterminal from the parse tree having the feature word as a head; (5) the entity type, as defined by the ACE guidelines; (6) the entity level (e.g. name); (7) the relation argument (e.g. REL-A); and (8) a WordNet hypernym. The new set of features F2 contains: (1) the predicate argument number provided by the semantic parser when using PropBank; (2) the predicate target for that argument; (3) the FE provided by the semantic parser when using FrameNet; (4) the target word for the frame; (5) the grammatical function; (6) the Frame; (7) the WordNet domain; (8) the WordNet concept that expressed the type of relation in which the word may belong within the domain; (9) a set of other related WordNet concepts (e.g. direct hypernym); (10) the ProbBank concepts that most frequently occur as arguments for the predicate; (11) the FrameNet concepts that most frequently occur in FEs for the frame. The features and their values for the node *"attorneys"* are listed in Figure 7.

Feature Set F1	Example	Feature Set F2	Example
1 Word	attorneys	1 Predicate−argument number	Arg0
2 Part−of−speech	NN	2 Predicate Target	represent
3 General−POS	noun	3 FE	Employee
4 Syntactic chunk tag	NP	4 Target word	represent
5 Entity−type	PERSON	5 Gramatical function	Ext
6 Entity−level	nominal	6 Frame	Employment
7 Relation−argument	REL−A	7 WordNet Domain	jurisprudence
8 WordNet hypernym	Individual	8 WordNet Relation Concept	lawyer−client
		9 WordNet Semantic Concepts	professional,paralegal
		10 PropBank(WN) Concepts	banker, lawyer, share
		11 FrameNet(WN) Concepts	relation, men

Figure 7: Sets of features assigned to each node in the dependency tree.

The features are used by a tree kernel function $K(T_1, T_2)$ that returns a similarity score in the range $(0, 1)$. We preferred the more general version of the kernel introduced in [1] to the kernel described by [8]. This kernel is based on two functions defined on the features of tree nodes: a matching function $m(t_i, t_j) \in \{0, 1\}$ and a similarity function $s(t_i, t_j) \in (0, \infty)$. The feature vector of a tree node $\phi(t_i) = \{v_1, \ldots v_d\}$ consists of two possibly overlapping subsets $\phi_m(t_i) \subseteq \phi(t_i)$ and $\phi_s(t_i) \subseteq \phi(t_i)$. As in [1], $\phi_m(t_i)$

are used by the matching function and $\phi_s(t_i)$ are used by the similarity function. The two functions are defined by:

$$m(t_i, t_j) = \begin{cases} 1 & \text{if } \phi_m(t_i) = \phi_m(t_j) \\ 0 & \text{otherwise} \end{cases}$$

and

$$s(t_i, t_j) = \sum_{v_q \in \phi_s(t_i)} \sum_{v_r \in \phi_s(t_j)} C(v_q, v_r)$$

where $C(v_q, v_r)$ is some compatibility function between the two feature values. For example, in the simplest case where

$$C(v_q, v_r) = \begin{cases} 1 & \text{if } v_q = v_r \\ 0 & \text{otherwise} \end{cases}$$

$s(t_i, t_j)$ returns the number of feature values in common between the feature vectors $\phi_s(t_i)$ and $\phi_s(t_j)$. For two dependency trees T_1 and T_2 with root nodes r_1 and r_2 the tree kernel is defined as:

$$K(T_1, T_2) = \begin{cases} 0 & \text{if } m(r_1, r_2)\text{=0} \\ s(r_1, r_2) + K_c(r_1[c], r_2[c]) & \text{otherwise} \end{cases}$$

where K_c is a kernel function over the children of the nodes r_1 and r_2. Let **a** and **b** be sequences of indices such that **a** is a sequence $a_1 \leq a_2 \leq \ldots \leq a_n$ and likewise for **b**. Let $d(a) = a_n - a_1 + 1$ and $l(a)$ be the length of **a**. Then for every $t_i \in T_1$ and $t_j \in T_2$, we have

$$K_c(t_i[c], t_j[c]) = \sum_{a,b,l(a)=l(b)} \lambda^{d(a)} \lambda^{d(b)} K(t_i[a], t_j[b])$$

where $\lambda \in (0, 1)$ represents a decay factor that penalizes matching subsequences that are spread out within the child sequences. The definition of K_c, the kernel function over children, assumes that the matching function used in the definition of the tree kernel $K(t_i[a], t_j[b])$ operates not only on single nodes, but also on node sequences $t_i[a]$ or $t_j[b]$. If all the nodes in the sequence are matched, $m(t_i[a], t_j[b]) = 1$. For each matching pair of nodes (a_i, b_j) is a matching subsequence, we accumulate the result of the similarity function $s(a_i, b_j)$ and then recursively search for matching subsequences of *their* children.

As in [1], we implemented two types of tree kernels: a contiguous kernel and a sparse kernel. A *contiguous* kernel only matches children subsequences that are uninterrupted by non-matching nodes. Therefore $d(a) = l(a)$. A *sparse* tree kernel, by contrast, allows non-matching nodes within matching subsequences.

4 Relation Extraction

When analyzing the dependency kernels, we noticed that only few nodes bear semantic information derived by the semantic parsers. We also noticed that these nodes are clustered together in the dependency tree. For example, Figure 6(d) illustrates the cluster of nodes from the dependency tree that contains semantic information. Instead of using the entire dependency tree to compute similarities, we selected sub-trees that contain nodes having values for the features from set F2 (illustrated in Figure 7). Typically these nodes correspond to target predicates and their arguments or FEs. This allowed us to compare trees of the form $SDT(R_1)$ [“$attorneys$”→“$represented$”←“$Woodward$”] and $SDT(R_2)$[“$intern$”→“$dismissed$”←“$lawyer$”]. We called such trees *semantic dependency trees* since they are characterized by semantic features present in the nodes of dependency trees. Semantic dependency trees (SDTs) are binary trees containing three nodes: a verbal predicate that is the root of the tree; and two children nodes, each an argument of the predicate. To measure the similarity of two SDTs we built a very simple kernel:

$$K(T_1, T_2) = \begin{cases} 0 & \text{if } m(c(T_1), c(T_2))\text{=0} \\ S(r(T_1), r(T_2)) + S_p(T_1, T_2) & \text{otherwise} \end{cases}$$

where the matching function is performed only on the children. The matching features that were used comprised ϕ_m^1=$\{General\text{-}POS,\ Entity\text{-}Type,\ Relation\text{-}argument\}$, ϕ_m^2=$\{FE\}$ and ϕ_m^3=$\{Predicate\text{-}argument\ number\}$. $m(c(T_1), c(T_2))$=1 if there is a combination of the pair of arguments that has the same matching features. For example, in the case of $SDT(R_1)$ and $SDT(R_2)$, the combination is {(“$attorneys$”, “$lawyer$”) and (“$intern$”, “$Woodward$”)} when using $\phi_m^1 \cup \phi_m^2$, since both attorney and lawyer are nouns, Persons, REL-A and they are both covered by the FE *Employee* in their respective frames. To measure the similarity between the verbal predicates, the function $S(r(T_1), r(T_2))$ measures the semantic compatibility of the predicates. The features used for measuring similarity of predicates are ϕ_S^P=$\{Frame,\ Predicate\ Target,\ WordNet\ Domain\}$. The compatibility measures assigned to each feature are: 1 if the same predicate target, 0.9 if both predicates are targets of the same frame, 0.7 if both predicates are targets of frames from the same event structure, 0.5 if both predicates belong to the same WordNet domain and 0.3 if the predicates are not covered in FrameNet, but have the same arguments in PropBank as other predicates from the same frame. The predicate similarity brings forward semantic frames that characterize a type of extraction relation. For example, in the case of Role_Client relation, such frames were (a) Employment_end and its subframes from the event structure; Employment_Start; and (b) Commerce_Sell or (c) Commerce_Buy.

The $S_p(T_1, T_2)$ similarity focuses on the combination of FEs or predicate-arguments that are identified for the children of the SDTs. The similarity features that are used are {FE, Predicate-argument number, Grammatical Function, WordNet Domain, WordNet Relation Concept}. For example, when the similarity $S_p(SDT(R_1), SDT(R_2))$ is computed, since we have an $\{Employer\_Employee\}$ relation between the FEs in both SDTs, the confidence assigned based on identical FE-FE relation is 1. If we have identical Predicate-argument numbers, the confidence is 0.6. For identical WordNet domains, 0.4 and for the same WordNet relation concept, we assign the confidence 0.7.

One limitation of the SDTs stems from the fact that this formalism cannot capture extraction relations that are expressed in the same noun phrase, e.g. “our customers”, “his urologist” or “George’s high school”. To recognize such relations we consider that they relate to some arguments of “unspecified” predicates. In the case when NPs contain pronouns, they are resolved by a successful coreference resolution algorithm [5], and the pronoun is substituted by a pair (referent,

CATEGORY). For example, the pronoun "our" from "our customers" is replaced by ("IBM", ORGANIZATION)

By measuring the semantic similarity between the pair (NP_head, Category) and any pair of arguments of a predicate from the training corpus, a plausible predicate can be found. For the noun phrase "our customers", the $SDT[\text{"}shop\text{"} \leftarrow \text{"}billed\text{"} \rightarrow \text{"}customers\text{"}]$ was deemed the most relevant, assigning the predicate "billed" to the two arguments: ORGANIZATION and "customers". The semantic similarity was measured by enhancing the WordNet, similarity measure is publicly available resource (*www.d.umn.edu/∼tpederse/Pubs/AAAI04PedersenT.pdf*), to handle semantic classes identified for entities, e.g. ORGANIZATION, DISEASE.

5 Experimental results

To evaluate the role of shallow semantics provided by semantic parsers on relation extraction we have used the Automatic Content Extraction (ACE) corpus available from LDC (LDC2003T11). The data consists of 422 annotated text documents gathered from various newspapers and broadcasts. Five entity types have been annotated (PERSON, ORGANIZATION, GEO-POLITICAL ENTITY, LOCATION, FACILITY) along with the 24 types of relations. The relations are listed in Figure 8.

At_Located	At_Residence	Role_Affiliate	Role_Founder
Role_Staff	Near_Relative−loc	Role_Client	Soc_Associate
Role_Member	Role_Citizen	Role_Owner	Soc_Spouse
Role_Mgmt	Soc_Professional	Role_Other	Soc_Sibling
Part_Part−of	Part_Subsidiary	Soc_Relative	Part_Other
At_Based−in	Soc_Parent	Soc_Personal	Soc_Grandparent

Figure 8: Extraction relations evaluated in ACE.

We implemented the same five kernels as [1]: K_0=sparse kernel, K_1=contiguous kernel, K_2=bag-of-words kernel and K_3=K_0+K_2 and K_4=K_1+K_2 and used first only the feature set F1 from Figure 7 and then both feature sets F1 and F2. The comparison of the kernel performance of the two experiments is listed in Figure 9.

Kernel	Avg. Precision		Avg. Recall		Avg. F−score	
	Features		Features		Features	
	F1	F1+F2	F1	F1+F2	F1	F1+F2
K0	52.3	55.8	19.7	34.6	28.62	42.71
K1	55.1	61.4	20.4	32.3	29.78	42.33
K2	38.2	48.7	7.2	28.5	12.12	35.96
K3	54.8	61.3	18.5	40.7	27.66	48.92
K4	60.5	72.2	20.3	44.5	30.4	55.06

Figure 9: Kernel performance comparison.

We used each kernel within an SVM (we augmented the SVMlight implementation to include our kernels). We choose to train on all 24 relations, not only on the first 5-high level relation types as was done in [1]. The results indicate that on average, for K_4, the best performing kernel, we obtained an increase of 24.66% in the F-score when features provided by the semantic parsers were added. When relying on SDTs, the average Precision that was obtained was 89.3%, the recall was 76.4%, thus an F-score of 82.35%, when using the same data as [1]. Figure 10 illustrates F-score obtained when using several other machine learning techniques available in the Weka package. In Figure 10 P indicates results for relations involving a predicate, NP indicates results for relations within an NP and C indicates the combined results. The results show that frame semantics produce an enhancement of 53.24% over previous state-of-art results in relation extraction. Furthermore, they show that semantic representations such as frames or predicate-argument structures have a wider impact on classification performance than the classification technique.

	Bayes Nets	Naive Bayes	AdaBoost	Bagging	Stacking	ID3	J48	Random Forest	Random Tree	SVM	Kernels on SDTs
P	77.97	79.10	62.14	82.48	62.14	73.45	83.05	82.48	79.09	81.35	82.17
NP	79.32	82.52	63.10	85.22	63.71	72.35	84.81	83.10	80.63	83.92	83.85
C	78.43	80.24	62.46	83.39	62.66	72.71	83.64	82.69	79.60	82.21	82.73

Figure 10: Results of relation extraction.

In the ACE data 61.71% of the training/testing data could be cast into SDTs. The semantic similarity between arguments of a relation within the same NP and arguments present in SDTs allowed the extraction with an average F1-score of 78.41%. The quality of the extraction results depend on the quality of the semantic parsers, that obtained F-scores of over 90% in recent SENSEVAL evaluations.

6 Conclusions

In this paper we have introduced a new dependency structure that relies on semantic information provided by shallow semantic parsers. This structure enabled the extraction of relevant relations with better performance than previous state-of-the-art kernel methods. Furthermore, the semantic features enabled similarly good results to be obtained with a few other learning algorithms. We also used compatibility functions that made use of semantic knowledge. This framework could be extended to allow processing of idiomatic predicates, e.g. [PERSON "lobbying on behalf of"" ORGANIZATION], and combined predications.

References

[1] A. Culotta and J. Sorensen. Dependency tree kernels for relation extraction. In *Proceedings of the ACL-2004*, 2004.

[2] C. Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press., 1998.

[3] Daniel Gildea and Daniel Jurasky. Automatic labeling of semantic roles. *Computational Linguistic*, 28(3):496–530, 2002.

[4] F. Jelineck, J. Lafferty, D. Magerman, R. Mercer, A. Ratnaparkhi, and S. Roukos. Decision tree parsing using a hidden derivational model. In *Proceedings of the HLT Workshop-1994*.

[5] X. Luo, A. Ittycheriah, H. Jing, N. Kambhatla, and S. Roukos. A Mention-Synchronous Coreference Resolution Algorithm Based On the Bell Tree. In *Proceedings of the ACL-2004*, 2004.

[6] S. Pradhan, K. Hacioglu, V. Krugler, W. Ward, J. H. Martin, and D. Jurafsky. Support Vector Learning for Semantic Argument Classifi cation. *Journal of Machine Learning Research*, 2004.

[7] M. Surdeanu, S. M. Harabagiu, J. Williams, and J. Aarseth. Using Predicate-Argument Structures for Information Extraction. In *Proceedings of the ACL-2003*, 2003.

[8] D. Zelenko, C. Aone, and A. Richardella. Kernel Methods for Relation Extraction. In *Proceedings of the EMNLP-2002*, pages 71–78, 2002.